# POSTER: Efficient approach to direct B–spline surface rendering by a ray tracing

Aleksandrs Sisojevs
Riga Technical University
Meza Str. 1/3 – 304
LV-1048, Riga, Latvia
alexiv@inbox.lv

Aleksandrs Glazs
Riga Technical University
Meza Str. 1/3 – 304
LV-1048, Riga, Latvia
glaz@egle.cs.rtu.lv

## ABSTRACT

There are different approaches to B-spline surface visualization by ray tracing methods. But these methods of computer graphics do not solve this task at an expected level. Therefore in this paper the authors propose to use a new approach for B-spline surface visualization by ray tracing method to render objects with high accuracy and quality.

## Keywords

B-spline surface, free-form surface, gradient methods, optimization, ray tracing, rendering.

## 1. INTRODUCTION

The ray tracing method is a popular technique for rendering high quality images. This paper presents a new approach for high-quality visualization of non-rational B-spline.

There are many approaches to solving this problem.

Martin et al. [Mar00a] describes a framework for ray-tracing of trimmed NURBS using Newton's method. A nonlinear equation system is used for finding the intersection point. The Newton's method is used for solving this equation system.

Nishita et al. [Nis90a] describes an iterative algorithm called Bézier Clipping, used to compute intersections between a ray and a Bézier patch by identifying and cutting away regions of the patch that are known not to intersect with the ray. There are several disadvantages in this approach, like unstable work for some surfaces and problems in solving the case of multiple equivalent intersection points.

Efremov et al. [Efr05a] proposed some modification to Nishita's algorithm for Bézier and NURBS surfaces. But in this case, the number of patches and calculations increases.



In this work we propose a new approach for finding the values of unknown parameters, which are necessary for visualization of a B - spline surface. The method is based on transforming the problem of finding the roots of equation to the task of optimization. In this case the optimization task is divided into two parts: preliminary search and optimization.

## 2. PROPOSED APPROACH

The mathematical task of finding an intersection point between the ray and a parametric surface can be described as a nonlinear equations system [Mar00a, Sis08a]:

$$\begin{cases} Q_X(u,v) = Q_X(t) \\ Q_Y(u,v) = Q_Y(t) \\ Q_Z(u,v) = Q_Z(t) \end{cases}, \quad (1)$$

where:

$Q_X(u,v)$, $Q_Y(u,v)$, $Q_Z(u,v)$, – surface equations,

$Q_X(t)$, $Q_Y(t)$, $Q_Z(t)$, – ray equations.

A non-rational B-spline surface can be formulated as [Rog90b, Hea04b]:

$$Q(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} P_{i,j} \cdot N_{i,p}(u) \cdot N_{j,q}(v), \quad (2)$$

where:

$P_{i,j}$ – control points,

$N_{i,p}(u)$, $N_{j,q}(v)$ – the B-spline polynomials;

 ISBN 978-80-86943-95-4

$u$, $v$ – parameters.

Analogically, a ray can be represented as:

$$Q(t) = \sum_{i=0}^{1} P_i \cdot N_{i,1}(t), \quad (3)$$

where:

$P_{i,}$ – control points,

$N_{i,1}(t)$ – the B-spline polynomials of the 1st power;

$t$ – parameter.

For calculation of unknown parameters $u$, $v$ and $t$, the above equation system should be considered as a case when the ray coincides with one of coordinate axes ($Oz$). In this case the system (1) is defined as follows [Sis08a]:

$$\begin{cases} Q_X(u,v) = 0 \\ Q_Y(u,v) = 0 \\ Q_Z(u,v) = Q_Z(t) \end{cases}, \quad (4)$$

There are many approaches for solving the equation system. In our work we transform the task of equation system solving to the optimization task. In order to find the parameters $u$ and $v$ only the first 2 equations should be considered, therefore the function of minimization can be described as follows:

$$w = \left(Q_X(u,v)\right)^2 + \left(Q_Y(u,v)\right)^2 \to \min_{u,v}, \quad (5)$$

We divided the task of function $w$ optimization into two parts: preliminary search and the optimization of the parameters (roots finding).

## 2.1 Preliminary search

In order to obtain the preliminary values of parameters $u$ and $v$ in every pixel, we propose to create a map of preliminary values. The map consists of the patch data and is coded in RGB channels. The Red channel contains the number of the patch ($r$=*number*+1). The Green and Blue channels contain the uniform gradient texture fill of the patches. Meaning that in the point $P_{0,0}$ the color value is ($r$;0;0), $P_{n,0}$ – ($r$;255;0), $P_{0,m}$ – ($r$;0;255) , $P_{n,m}$ – ($r$;255;255) and the rest of the gradient texture color values are evenly distributed on the patch surface.

In this case the preliminary values of parameters can be described as is follows:

$$\begin{cases} u_0 = (u_{\max} - u_{\min}) \cdot \dfrac{g}{255} + u_{\min} \\ v_0 = (v_{\max} - v_{\min}) \cdot \dfrac{b}{255} + v_{\min} \end{cases}, \quad (6)$$

The map is coded using OpenGL graphics library.

The example of preliminary values map is shown in Figure 1.

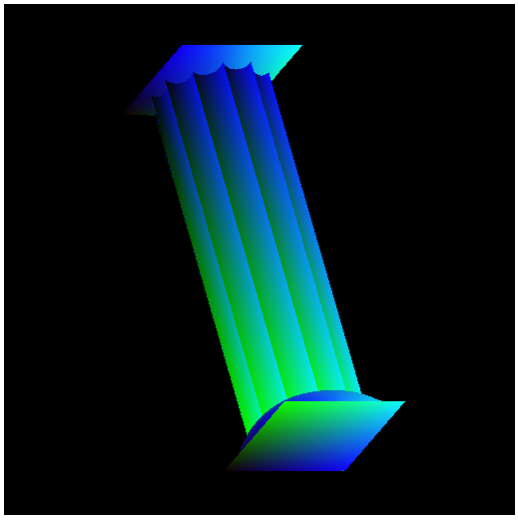

**Figure 1. The example of preliminary values map.**

## 2.2 Root finding

For optimization we chose a gradient method [Him72b]. The values for these parameters are defined at each search step as follows [Sis08a]:

$$\begin{cases} u^{new} = u^{old} - h\dfrac{\partial w}{\partial u} \\ v^{new} = v^{old} - h\dfrac{\partial w}{\partial v} \end{cases}, \quad (7)$$

where: $h$ – weight working step.

In matrix form this equation system (7) can be formulated as is follows:

$$\begin{bmatrix} u^{new} \\ v^{new} \end{bmatrix} = \begin{bmatrix} u^{old} \\ v^{old} \end{bmatrix} - 2 \cdot h \cdot \begin{bmatrix} \dfrac{\partial Q_X(u,v)}{\partial u} & \dfrac{\partial Q_Y(u,v)}{\partial u} \\ \dfrac{\partial Q_X(u,v)}{\partial v} & \dfrac{\partial Q_Y(u,v)}{\partial v} \end{bmatrix} \cdot \begin{bmatrix} Q_X(u,v) \\ Q_Y(u,v) \end{bmatrix}, \quad (8)$$

The optimization is carried out until the following condition is met:

$$w \le \varepsilon, \quad (9)$$

where ε – is a small number (in this paper we assume ε<<0,0001).

 ISBN 978-80-86943-95-4

## 2.3 Fast root finding (modification)

The approach described in section 2.2 is effective for the task of optimization. However, such approach gives an overflow of calculations. This is connected to the necessity of calculating a gradient during each iteration step. At this time it is possible to use the previous direction of a gradient (in case of the function *w* value decrease). In a pseudo-code it can be described as follows:

$w_{old}=w(u_{old}, v_{old})$

$u_{new}=u_{old}-u_{grad}$

$v_{new}=v_{old}-v_{grad}$

$w_{new}=w(u_{new}, v_{new})$

if $w_{new}>w_{old}$ then

(

$u_{grad}=grad\_u(u_{old}, v_{old})$

$v_{grad}=grad\_v(u_{old}, v_{old})$

$u_{new}=u_{old}-u_{grad}$

$v_{new}=v_{old}-v_{grad}$

)

In order to find an intersection point of a surface with any ray it is sufficient to transform such a ray and the control points of a surface with the help of elementary geometrical transformations in such a way that the ray would coincide with a necessary coordinate axis (in our example - axis Z).

# 3. EXPERIMENTAL RESULTS

In this work the proposed method, as well as the methods suggested by Martin et al were implemented.

In order to visualize a scene the 1 ray/pixel approach was used. The size of the images in experiment is 512*512 pixels. The experiments were carried out on the computer with CPU Intel Xeon 3,2 GHz, RAM 2 GB. The received images are shown in a Figure 2-7.

As it is possible to see from these figures the proposed methods gives an advantage on quality of the images (there's no distortion on the borders of patches). Image rendering time is shown in Table 1.

| Method | "Column" | "Water-lily" |
|---|---|---|
| Proposed | 74,906 | 180,234 |
| Proposed (modif.) | *31,156* | *98,313* |
| Martin et al. | 60,750 | 205,600 |

**Table 1. Image rendering time in seconds**

As seen from the table the unmodified proposed method gives unstable results on rendering time (compared to method Martin et all.). While, the modified proposed method results in the fastest rendering time in our experiments.

For comparison we shall consider the time of visualization in percentage, by taking earlier known method (Martin et al) for 100%. The results are shown in Table 2. As seen from the table, the modified proposed method works 2 times faster than the other methods.

| Method | "Column" | "Water-lily" |
|---|---|---|
| Proposed | 123,3 | 87,7 |
| Proposed (modif.) | *51,3* | *47,8* |
| Martin et al. | 100 | 100 |

**Table 2. Images rendering time in percentage (Martin et al. – 100%)**

# 4. CONCLUSION

The results (in Figures 2 – 7) seen, that:

- As seen from comparison of Figure 2 (or 3) with Figure 4, and Figure 5 (or 6) with Figure 7, the use of the modified proposed method results in better quality of the image than Martin et al. method. The proposed method has no distortions on the image, while method Martin et al has some faults (like distortion on border of patches).
- The modified proposed method results in faster image rendering time. This method works approximately 2 times faster than other methods.

# 5. REFERENCES


[Nis90a] Nishita. T., Sederberg. T.W., and Kakimoto M. Ray tracing trimmed rational surface patches, Computer Graphics, vol.24, no. 4, pp. 337–345, 1990.

[Efr05a] Efremov, A., Havran, V. & Seidel, H.-P. 'Robust and numerically stable Bézier clipping method for ray tracing NURBS surfaces.' In Proceedings of the 21st Spring Conference on Computer Graphics SCCG'05:127 - 135. 2005

[Mar00a] Martin, W., Cohen, E., Fish, R., & Shirley, P. (2000). 'Practical Ray Tracing of Trimmed NURBS Surfaces.' JGT 5(1): 27-52. 2000.

[Rog90b] Rogers, D.F. and Adams, J.A. Mathematical Elements for Computer Graphic, 2nd Ed., McGraw-Hill, Boston, MA, 1990..

[Hea04b] Hearn, D. and Baker, M.P. Computer Graphics with OpenGL, 3rd Ed., Prentice Hall, 2004.

[Him72b] Himmelblau, D. Applied Nonlinear Programming, – McGraw-Hill Book Company, 1972.

[Sis08a] Sisojevs A, Glazs A An new approach of visualization of free-form surfaces by a ray tracing method, IEEE MELECON Proceedings, 2008, pp 872-875

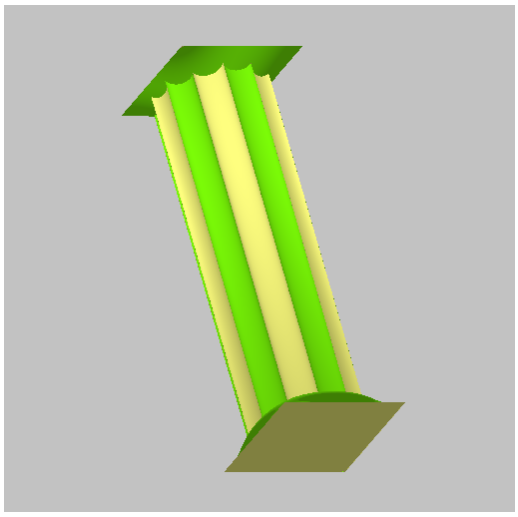

**Figure 2. The image of an object obtained using the proposed method.**

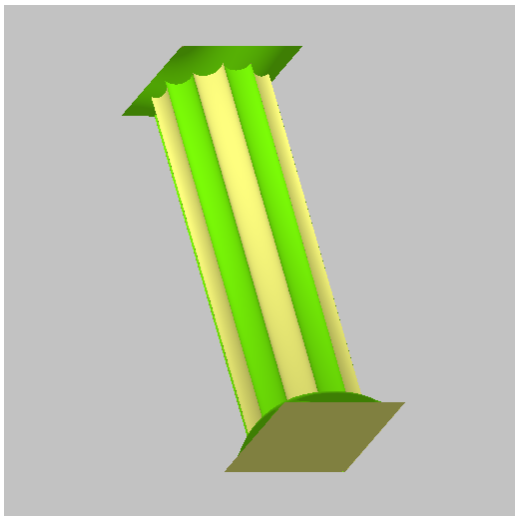

**Figure 3. The image of an object obtained using the proposed method (modification).**

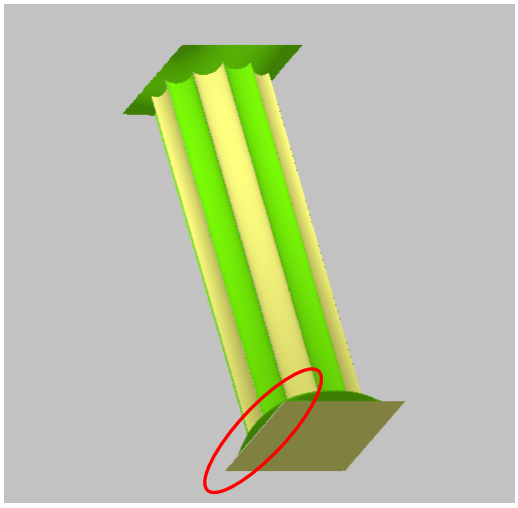

**Figure 4. The image of an object obtained using the method Martin et al.**

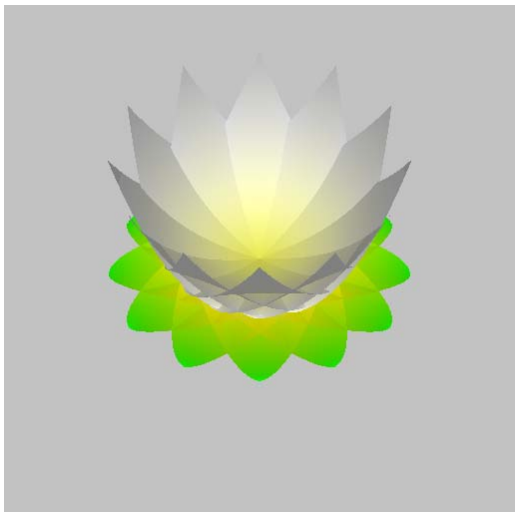

**Figure 5. The image of an object obtained using the proposed method.**

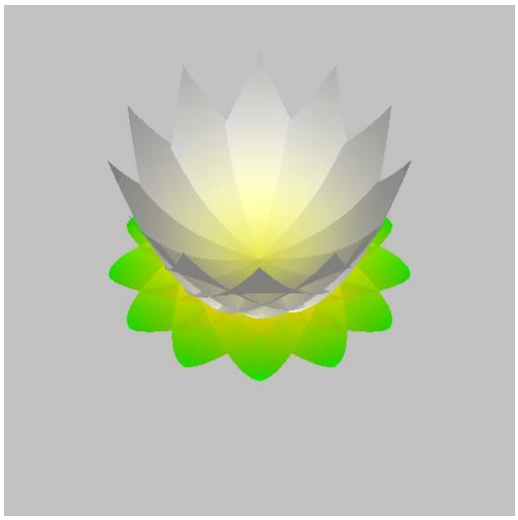

**Figure 6. The image of an object obtained using the proposed method (modification).**

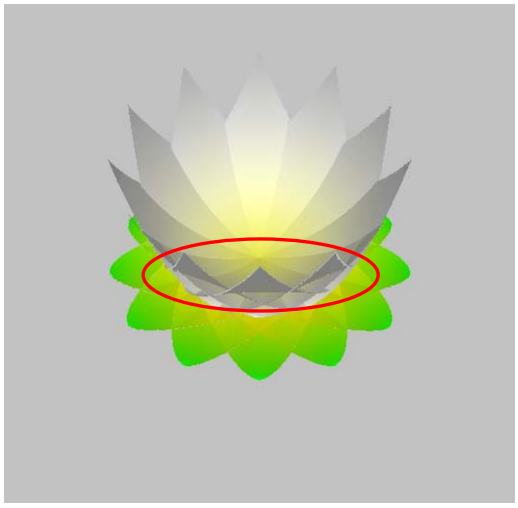

**Figure 7. The image of an object obtained using the method Martin et al.**